Docket No.: 42P15764 Confirmation No. 6169 Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BOARD OF PATENT APPEALS AND INTERFERENCES

In Re Application of: William W. Macy JR. et al. Application No. 10/612,592 Filed: July 1, 2003 For: METHOD AND APPARATUS FOR PARALLEL TABLE LOOKUP USING SIMD INSTRUCTIONS	Examiner: David H. Malzahn Art Unit: 2123 **CERTIFICATE OF TRANSMISSION** I hereby certify that this correspondence is being transmitted to the United States Patent and Trademark Office, or facsimile transmitted to Fax No. (531)273-8300 on 8-27-2007 /Lawrence M. Mennemeier/ Date Lawrence M. Mennemeier

APPELLANT'S BRIEF UNDER 37 CFR § 41.37
IN SUPPORT OF APPELLANT'S APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Mail Stop Appeal Brief-Patents
Commissioner of Patents
PO Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Appellant hereby submits this Brief in support of an appeal from a final decision of the Examiner, in the above-referenced case. Appellant respectfully requests consideration of this appeal by the board of Patent Appeals and Interference for allowance of the above-referenced patent application.

TABLE OF CONTENTS

I. Real Party in Interest

The real party in interest in the present appeal is Intel Corporation of Santa Clara, California, the assignee of the present application.

II. Related Appeals and Interferences

There are no related appeals or interferences to appellant's knowledge that would have a bearing on any decision of the Board of Patent Appeals and Interferences.

III. Status of the Claims (independent claims shown in bold)

Claims **1**-40 are pending in the application.

Claims **1**-9, **10**-19, **20-27** and **28**-34 stand rejected under 35 USC § 101 as allegedly being directed to a program per se.

Claims **35**-40 are allowed.

Final rejection of claims **1**-9, **10**-19, **20**-27 and **28**-34 is being appealed.

IV. Status of Amendments

An official amendment and response to a first Office Action mailed 8/17/2006 was submitted by appellant on 2/20/2007 and was entered. A Final Office Action was mailed on 3/26/2007. A Notice of Appeal was transmitted on 6/26/2007, and an appeal ensued.

Accordingly, the claims stand as of appelant's response of 2/20/2007, and are reproduced in clean form in the Claims Appendix.

V. Summary of Claimed Subject Matter

Appellant's disclosure describes methods, apparatus, and program means for performing parallel table lookup using SIMD (single instruction multiple data) instructions for performing operations on packed data. The method of one embodiment comprises loading a table having a set of L data elements. A determination is made whether the table fits into a single register. If the determination indicates that the table does, in fact, fit into a single register, a data lookup into the table is performed with a packed data shuffle operation. If the table does not fit into a single register, the table is divided into sections. Each of the sections is sized to fit into a single register. Packed data shuffle operations are then executed on these sections to look up data in the table.

Claim 1, for example, sets forth a method comprising: loading a table having a set of L data elements[1]; determining whether said table fits into a single register[2]; performing a data lookup into said table with a packed data shuffle operation if said determination

[1] "At block 1202, a table having a plurality of data elements are loaded." **(US 2004/0054879 A1 p. 15, par. 122, lines 2-3; Fig. 12, 1202)**

[2] "A determination is made at block 1204 as to whether the table fits in a single register." **(US 2004/0054879 A1 p. 15, par. 122, lines 3-5; Fig. 12, 1204)**

indicates that said table does fit into a single register[3]; and dividing said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and executing a plurality of packed data shuffle operations on said plurality of sections to look up data in said table[4].

Similarly, claim 20 sets forth an article comprising a tangible machine readable medium that stores a program, said program being executable by a machine[5,6] to perform a method comprising: determining whether a table having a set of L data elements fits into a single register[2]; performing a data lookup into said table with a packed data shuffle

---

[3] "If the table fits into a single register, the table lookup is performed with a shuffle operation at block 1216." **(US 2004/0054879 A1 p. 15, par. 122, lines 5-7; Fig. 12, 1216)**

[4] "One embodiment of a method for handling oversized tables divides a table into sections, each equal to the capacity of a register, and accesses each of these table sections with a shuffle instruction." **(US 2004/0054879 A1 p. 12, par. 109, lines 12-16)** "If the data does not fit into a single register, table lookup is to be performed with shuffle operations for each relevant portion of the table at block 1206." **(US 2004/0054879 A1 p. 15, par. 122, lines 7-9; Fig. 12, 1206)**

[5] "As processor technology advances, newer software code is also being generated to run on machines with these processors. Users generally expect and demand higher performance from their computers regardless of the type of software being used. One such issue can arise from the kinds of instructions and operations that are actually being performed within the processor. Certain types of operations require more time to complete based on the complexity of the operations and/or type of circuitry needed. This provides an opportunity to optimize the way certain complex operations are executed inside the processor." **(US 2004/0054879 A1 p. 1, par. 5)** "In an embodiment, the methods of the present invention are embodied in machine-executable instructions. The instructions can be used to cause a general-purpose or special-purpose processor that is programmed with the instructions to perform the steps of the present invention." **(US 2004/0054879 A1 p. 2, par. 33, lines 1-5)** "Although the below examples describe instruction handling and distribution in the context of execution units and logic circuits, other embodiments of the present invention can be accomplished by way of software. The present invention may be provided as a computer program product or software which may include a machine or computer-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. Such software can be stored within a memory in the system." **(US 2004/0054879 A1 p. 2, par. 34, lines 1-11)** "Accordingly, the computer-readable medium includes any type of media/machine-readable medium suitable for storing or transmitting electronic instructions or information in a form readable by a machine (e.g., a computer)." **(US 2004/0054879 A1 p. 2, par. 35, lines 1-5)**

[6] "Memory **120** can be a dynamic random access memory (DRAM) device, a static random access memory (SRAM) device, flash memory device, or other memory device. Memory **120** can store instructions and/or data represented by data signals that can be executed by the processor **102**." **(US 2004/0054879 A1 p. 4, par. 51, lines 4-9, Fig. 1A, 120)** "The instruction prefetcher **226** fetches macro-instructions from memory and feeds them to an instruction decoder **228** which in turn decodes them into primitives called micro-instructions or micro-operations (also called micro op or uops) that the machine know how to execute. The trace cache **230** takes decoded uops and assembles them into program ordered sequences or traces in the uop queue **234** for execution. When the trace cache **230** encounters a complex macro-instruction, the microcode ROM **232** provides the uops needed to complete the operation." **(US 2004/0054879 A1 p. 6, par. 64, lines 16-26; Fig. 2, 230)**

operation if said determination indicates that said table does fit into a single register[3]; and dividing said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and executing a plurality of packed data shuffle operations on said plurality of sections to look up data in said table[4].

Claim 28 further sets forth an apparatus comprising: an execution unit[7] to execute a sequence of instructions[6], said instructions to perform a table lookup operation, said instructions to cause said execution to: determine whether a table having a set of data elements fits into a single register[2]; perform a data lookup into said table with a packed data shuffle operation if said determination indicates that said table does fit into a single register[3]; and divide said table into a plurality of sections if said table does not fit into a

---

[7] "**Figure 1A** is a block diagram of a computer system **100** formed with a processor **102** that includes one or more execution units **108** to perform a data shuffle algorithm in accordance with the present invention." **(US 2004/0054879 A1 p. 4, par. 48, lines 1-4, Fig. 1A, 108)** "Execution unit **108**, including logic to perform integer and floating point operations, also resides in the processor **102**. The processor **102** also includes a microcode (ucode) ROM that stores microcode for certain macroinstructions. For this embodiment, execution unit **108** includes logic to handle a packed instruction set **109**. In one embodiment, the packed instruction set **109** includes a packed shuffle instruction for organizing data." **(US 2004/0054879 A1 p. 4, par. 50, lines 1-8, Fig. 1A, 108)** "Alternate embodiments of an execution unit **108** can also be used in micro controllers, embedded processors, graphics devices, DSPs, and other types of logic circuits." **(US 2004/0054879 A1 p. 4, par. 51, lines 1-3, Fig. 1A, 108)** "Processing core **159** comprises an execution unit **142**, a set of register file(s) **145**, and a decoder **144**. Processing core **159** also includes additional circuitry (not shown) which is not necessary to the understanding of the present invention. Execution unit **142** is used for executing instructions received by processing core **159**. In addition to recognizing typical processor instructions, execution unit **142** can recognize instructions in packed instruction set **143** for performing operations on packed data formats. Packed instruction set **143** includes instructions for supporting shuffle operations, and may also include other packed instructions." **(US 2004/0054879 A1 p. 5, par. 57, lines 1-12, Fig. 1B, 142)** "For one embodiment, SIMD coprocessor **161** comprises an execution unit **162** and a set of register file(s) **164**. One embodiment of main processor **165** comprises a decoder **165** to recognize instructions of instruction set **163** including SIMD shuffle instructions for execution by execution unit **162**." **(US 2004/0054879 A1 p. 5-6, par. 61, lines 1-6, Fig. 1C, 162)** "For one embodiment of processing core **170**, main processor **166**, and a SIMD coprocessor **161** are integrated into a single processing core **170** comprising an execution unit **162**, a set of register file(s) **164**, and a decoder **165** to recognize instructions of instruction set **163** including SIMD shuffle instructions." **(US 2004/0054879 A1 p. 6, par. 63, lines 10-16, Fig. 1C, 162)** "The execution block **211** contains the execution units **212**, **214**, **216**, **218**, **220**, **222**, **224**, where the instructions are actually executed. This section includes the register files **208**, **210**, that store the integer and floating point data operand values that the micro-instructions need to execute. The processor **200** of this embodiment is comprised of a number of execution units: address generation unit (AGU) **212**, AGU **214**, fast ALU **216**, fast ALU **218**, slow ALU **220**, floating point ALU **222**, floating point move unit **224**. For this embodiment, the floating point execution blocks **222**, **224**, execute floating point, MMX, SIMD, and SSE operations." **(US 2004/0054879 A1 p. 7, par. 69, lines 1-12, Fig. 2, 222, 224)** "The data operands **510**, **520**, are sent to the shuffle logic **530** of an execution unit in the processor along with a shuffle instruction." **(US 2004/0054879 A1 p. 9, par. 86, lines 2-5, Fig. 5, 530)**

single register, each of said sections sized to fit into a single register, and execute a plurality of packed data shuffle operations on said plurality of sections to look up data in said table[4].

The method of an alternative embodiment comprises loading two M-bits wide data portions and shuffling both in accordance with an M-bits wide mask to generate shuffled results, then merging selected data elements of the shuffled results to accomplish an M-bits wide table lookup.

Claim 10, for example, sets forth an alternative method for table lookup comprising: loading data for a first M-bits wide portion[8] and data for a second M-bits wide portion of a table[9]; loading an M-bits wide mask, said mask comprised of N control elements, each control element corresponding to a unique data element position[10]; shuffling said first M-bits wide portion in accordance to said M-bits wide mask to generate a first shuffled result[4,11]; shuffling said second M-bits wide portion in accordance to said M-bits wide mask to generate a second shuffled result[4,12];

---

[8] "This first set of data elements is grouped as an operand named LOW TABLE DATA **1021**." **(US 2004/0054879 A1 p. 13, par. 111, lines 3-4, Fig. 10A, 1021)** "At block **1104**, the data elements for a first portion of a table or a first data set is loaded." **(US 2004/0054879 A1 p. 14, par. 120, lines 10-12, Fig. 11, 1104)**

[9] "This second set of data elements is grouped as an operand named HIGH TABLE DATA **1051**." **(US 2004/0054879 A1 p. 13, par. 111, lines 16-18, Fig. 10B, 1051)** "Data elements for a second portion of a table or a second data set is loaded at block **1108**." **(US 2004/0054879 A1 p. 14, par. 120, lines 14-15, Fig. 11, 1108)**

[10] "MASK **1001** and LOW TABLE DATA **1021** are each comprised of sixteen elements in this example." **(US 2004/0054879 A1 p. 13, par. 111, lines 4-6, Figs. 10A-B, 1001)** "Because the same set of masks **1001** were used with both the LOW TABLE DATA **1021** and HIGH TABLE DATA **1051**, their respective resultants **1041**, **1042**, appear to have similarly positioned data, but from different source data." **(US 2004/0054879 A1 p. 13, par. 112, lines 1-5, Figs. 10A-B, 1001)** "At block **1102**, a set of shuffle masks designating a shuffle pattern is received." **(US 2004/0054879 A1 p. 14, par. 120, lines 6-8, Fig. 11, 1102)**

[11] "A shuffle operation of MASK **1001** and LOW TABLE DATA **1021** yields a resultant TEMP RESULTANT A **1041**." **(US 2004/0054879 A1 p. 13, par. 111, lines 6-8, Fig. 10A, 1041)** "The first portion data elements are shuffled in accordance to the shuffle pattern of block **1102** at block **1106**." **(US 2004/0054879 A1 p. 14, par. 120, lines 12-14, Fig. 11, 1106)**

[12] "A shuffle operation of MASK **1001** and HIGH TABLE DATA **1051** yields a resultant TEMP RESULTANT B **1042**." **(US 2004/0054879 A1 p. 13, par. 111, lines 19-21, Fig. 10B, 1042)** "The second portion data elements are shuffled in accordance to the shuffle pattern of block **1102** at block **1110**." **(US 2004/0054879 A1 p. 14, par. 120, lines 15-17, Fig. 11, 1110)**

and merging selected data elements from said first and second shuffled results to obtain an M-bits wide table lookup resultant[13].

[13] "**Fig. 10H** illustrates the merging of the selected data from the first data set and the second data set. A packed logical OR operation is performed on SELECTED LOW TABLE DATA **1049** and SELECTED HIGH TABLE DATA **1050** to obtain MERGED SELECTED TABLE DATA **1070**, which is the desired resultant of the parallel table lookup algorithm in this example. In an alternative embodiment, a packed addition operation to add together SELECTED LOW TABLE DATA **1049** and SELECTED HIGH TABLE DATA **1050** can also yield MERGED SELECTED TABLE DATA **1070**." **(US 2004/0054879 A1 p. 14, par. 118, lines 1-11, Fig. 10C-H, 1070)**
"At block **1112**, table selects are filtered out from the shuffle masks. The table selects of this embodiment involve the source select fields that designate where a data element is supposed to originate from. At block **1114**, a table select mask is generated for the shuffled data from the first portion of the table. A table select mask is generated for the shuffled data from the second portion of the table at block **1116**. These table select masks are to filter out the desired shuffled data elements for specific data element positions from the appropriate table data source.
At block **1118**, data elements are selected from the shuffled data of the first table portion in accordance with a table select mask of block **1114** for the first table portion. Data elements are selected at block **1120** from the shuffled data of the second table portion in accordance with the table select mask of block **1116** for the second table portion. The shuffled data elements selected from the first table portion at block **1118** and from the second table portion at block **1120** are merged together at block **1122** to obtain merged table data. The merged table data of one embodiment includes data elements shuffled from both the first table data and the second table data. For another embodiment, the merged table data can include data looked up from more than two table sources or memory regions." **(US 2004/0054879 A1 p. 14-15, par. 120, line 17 throught par. 121 line 14, Fig. 11, 1112-1122)**

VI. Grounds of Rejection to be Reviewed on Appeal

A. Claims **1**-9, **10**-19, **20**-27 and **28**-34 stand rejected under 35 USC § 101 as allegedly being directed to a program per se.

VII. Argument

A. 35 U.S.C. § 101 REJECTIONS

Claims **1**-9, **10**-19, **20**-27 and **28**-34 stand rejected under 35 USC § 101 as allegedly being directed to a program per se. The Final Office Action (p. 2, lines 18-20) states that since the claimed invention can be accomplished by way of software, it reads on a program per se.

Appellant respectfully submits that, if the examiner's assertion were true, then any process that could be automated through programmed machines would be non-statutory as a program per se. But it may very well be the case that any process which can be listed as steps in a claim can in fact be automated by programmed machines. Therefore, according to the examiner's assertion, any process which can be listed as steps in a claim is not patentable.

To this point, the Federal Circuit explained in *AT & T Corp. v. Excel Communications, Inc.* 172 F.3d 1352, 1356, 50 USPQ2D 1447, 1450 (Fed. Cir. 1999) that (emphasis added):

> This court recently pointed out that any step-by-step process, be it electronic, chemical, or mechanical, involves an "algorithm" in the broad sense of the term. *See State Street Bank & Trust Co. v. Signature Fin. Group, Inc.*, 149 F.3d 1368, 1374-75, 47 USPQ2d 1596, 1602 (Fed. Cir. 1998).

Yet 35 U.S.C. §101 clearly sets forth that (emphasis added), "Whoever invents or discovers <u>any new and useful process</u>, machine, manufacture, or composition of matter, or any new and useful improvement thereof, <u>may obtain a patent therefor</u>, subject to the conditions and requirements of this title."

Thus, contrary to the examiner's assertion, the intent of such broad coverage for "any new and useful process" does not prohibit some particular new and useful processes simply because they can be performed by, or make use of a computer program.

Therefore, appellant respectfully submits that the Examiner is in error for incorrectly concluding that the processes of claims 1 and 10, the executable program on a tangible machine readable medium of claim 20, and the programmed apparatus of claim 28 are each directed to programs per se (i.e. programs—of, in or by themselves, their functionally descriptive material standing alone without reference to additional means which would allow said functionality to be realized). Indeed, the MPEP §2106.01 further states that (emphasis added):

> Computer programs are often recited as part of a claim. USPTO personnel should <u>determine whether the computer program is being claimed as part of an otherwise statutory manufacture or machine</u>. In such a case, the claim remains statutory irrespective of the fact that a computer program is included in the claim. The same result occurs when a computer program is used in a computerized process where the computer executes the instructions set forth in the computer program. <u>Only when the claimed invention taken as a whole is directed to a mere program listing, i.e., to only its description or expression, is it descriptive material *per se*</u> and hence nonstatutory.

Since the instant claims are not directed to a mere program listing but rather to otherwise statutory processes, articles and apparatus, the question at hand, is whether such processes are "new and useful."

An analysis of the instant claims must be performed in order to make a determination of whether the subject matter is statutory. Such analysis should correlate

each claim element with corresponding structures, materials or acts set forth in the specification.

Claim 1, for example, sets forth:

> 1. (Original) A method comprising:
> loading a table having a set of L data elements; determining whether said table fits into a single register;
> performing a data lookup into said table with a packed data shuffle operation if said determination indicates that said table does fit into a single register; and
> dividing said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and executing a plurality of packed data shuffle operations on said plurality of sections to look up data in said table.

The Federal Circuit makes it clear that the ordinary and customary meaning of a claim term is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention. "The person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification." *Phillips v. AWH Corp.*, 415 F.3d at 1313.

Appellant respectfully submits that the instant language, when correlated with the corresponding structures and processes set forth in the specification (e.g. see Figures 10A-H, 11 and 12; paragraphs 110-122) makes it apparent to one of skill in the art that the claimed invention has practical applications in the technical arts, i.e. to order data from small lookup tables such as may be used in video and encryption applications.

In addition, Applicant respectfully submits, that the present application clearly asserts such a practical application in the technical arts.

For example, paragraph 39 of the specification (emphasis added) asserts that:

> Embodiments of the present invention provide a way to implement a packed byte shuffle instruction with a flush to zero capability as an algorithm that makes use of SIMD related hardware. For one embodiment, the algorithm is based on the concept of shuffling data from a particular register or memory location based on the values of a control mask for each data

element position. Embodiments of a packed byte shuffle can be used to reduce the number of instructions required in many different applications that rearrange data. A packed byte shuffle instruction can also be used for any application with unaligned loads. Embodiments of this shuffle instruction can be used for filtering to arrange data for efficient multiply-accumulate operations. Similarly, a packed shuffle instruction can be used in video and encryption applications for ordering data and small lookup tables. This instruction can be used to mix data from two or more registers. Thus embodiments of a packed shuffle with a flush to zero capability algorithm in accordance with the present invention can be implemented in a processor to support SIMD operations efficiently without seriously compromising overall performance.

Paragraph 45 of the specification (emphasis added) asserts that:

> Similarly, the reversing of all the bytes in a 128 bit register, such as in changing between big endian and little endian formats, can be easily performed with a single packed shuffle instruction. Whereas even these fairly simple patterns require a number of instructions if a packed shuffle instruction were not used, complex or random patterns require even more inefficient instruction routines. The most straight forward solution to rearrange random bytes in a SIMD register is to write them to a buffer and then use integer byte reads/writes to rearrange them and read them back into a SIMD register. All these data processing would require a lengthy code sequence, while a single packed shuffle instructions can suffice. By reducing the number of instructions required, the number of clock cycles needed to produce the same result is greatly reduced. Embodiments of the present invention also use shuffle instructions to access multiple values in a table with a SIMD instructions. Even in the case where the a table is twice the size of a register, algorithms in accordance with the present invention allow for accesses to data elements at a faster rate than the one data element per instruction as with integer operations.

Paragraphs 108-109 of the specification (emphasis added) assert that:

> Currently, table lookups using integer instructions requires a large number of instructions. An even greater number of instructions are needed per lookup if integer operations are used to access data for algorithms implemented with SIMD instructions. But by using embodiments of a packed byte shuffle instruction, the instruction count and execution time is drastically reduced. For instance, sixteen data bytes can be accessed during a table lookup with a single instruction if the table size is sixteen bytes or less. Eleven SIMD instructions can be used to lookup table data if the table size is between seventeen and thirty two bytes. Twenty three SIMD instructions are needed if the table size is between thirty three and sixty four bytes.
>
> There are some applications with data parallelism that cannot be implemented with SIMD instructions due to their use of lookup tables. The quantization and deblocking algorithms of the video compression method H.26L is an example of an algorithm that uses small lookup tables that may not fit into a 128 bit register. In some cases, the lookup tables used by these algorithms are small. If the table can fit in a single register, the table lookup operation can be accomplished with one packed shuffle instruction. But if the memory space requirement of the table exceeds the size of a single register, embodiments of a packed shuffle instruction can still work via a different algorithm. One embodiment of a method for handling oversized tables divides a table into sections, each equal to the capacity of a register, and accesses each of these table sections with a shuffle instruction. The shuffle instruction uses the same shuffle control sequence to access each section of the table. As a result, a parallel table lookup can be implemented in these cases with the packed byte shuffle instruction, thus permitting the use of SIMD instructions to improve algorithm performance. Embodiments of the present invention can help improve performance and reduce the number of memory accesses needed for algorithms that use small lookup tables. Other embodiments also permit access of multiple

lookup table elements using SIMD instructions. A packed byte shuffle instruction in accordance to the present invention permits efficient SIMD instruction implementation instead of less efficient integer implantation of algorithms that use small lookup tables. This embodiment of the present invention demonstrates how to access data from a table that requires memory space larger than a single register. In this example, the registers contain different segments of the table

Paragraph 147 of the specification (emphasis added) asserts that:

> Embodiments of algorithms using packed shuffle instructions in accordance with the present invention can also improve processor and system performance with present hardware resources. But as technology continues to improve, embodiments of the present invention when combined with greater amounts of hardware resources and faster, more efficient logic circuits, can have an even more profound impact on improving performance. Thus, one efficient embodiment of a packed shuffle instruction having byte granularity and a flush to zero option can have different and greater impact across processor generations. Simply adding more resources in modem processor architectures alone does not guarantee better performance improvement. By also maintaining the efficiency of applications like one embodiment of the parallel table lookup and the packed byte shuffle instruction (PSHUFB), larger performance improvements can be possible.

Thus the specification makes it readily apparent to one of skill in the art that the claimed invention has a practical application in the technical arts.

The Supreme Court held that the focus in any statutory subject matter analysis be on the claim as a whole, stating "When a claim containing a mathematical formula implements or applies that formula in a structure or process which, when considered as a whole, is performing a function which the patent laws were designed to protect (e.g., transforming or reducing an article to a different state or thing, then the claim satisfies the requirements of § 101." *In re Alappat*, 33 F.3d 1526, 1543 (Fed. Cir. 1994) (quoting *Diehr*, 450 U.S. at 192, 209 USPQ at 10).

This notion is sometimes phrased in terms of requiring a transformation or reduction of 'subject matter.' In Schrader, the phrase 'subject matter' was determined not to be limited to tangible articles or objects, but includes intangible subject matter, such as data or signals (e g. encrypted and/or video communication), representative of or

constituting physical activity or objects. *Schrader*, 22 F.3d at 295, 30 USPQ2D (BNA) at 1459.

Thus appellant respectfully submits that Claims 1-34 are directed to statutory subject matter.

Conclusion

Appellant submits that all claims now pending are in condition for allowance. Such action is earnestly solicited at the earliest possible date. If there is a deficiency in fees, please charge our Deposit Acct. No. 50-0221.

Respectfully submitted,

Date: August 27, 2007

/s/Lawrence M. Mennemeier/
Lawrence M. Mennemeier
Reg. No. 51,003

INTEL CORPORATION
c/o INTELLEVATE LLP
P.O. Box 52050
Minneapolis, MN 55402
(408) 765-2194

VIII. Claims Appendix: Claims Allowed and Involved in Appeal (Clean Copy)

1. (Original) A method comprising:

loading a table having a set of L data elements; determining whether said table fits into a single register;

performing a data lookup into said table with a packed data shuffle operation if said determination indicates that said table does fit into a single register; and

dividing said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and executing a plurality of packed data shuffle operations on said plurality of sections to look up data in said table.

2. (Original) The method of claim 1 further comprising loading a lookup mask for each packed data shuffle operation, said lookup mask to indicate which data elements are to be extracted from said table.

3. (Original) The method of claim 2 wherein said lookup mask is comprised of L shuffle masks, each shuffle mask corresponding to a unique data element position.

4. (Original) The method of claim 3 wherein each shuffle mask is comprised of: a flush to zero field, said flush to zero field to indicate whether a data element position associated with this shuffle mask is to be filled with a zero value; a selection field, said

selection field to indicate which table data element to shuffle data from; and a source select field, said source select field to indicate which of said plurality of table sections to shuffle data from for this shuffle mask.

5. (Original) The method of claim 2 further comprising merging shuffle results from said plurality of packed data shuffle operations into a single register.

6. (Original) The method of claim 3 wherein each packed shuffle operation comprises: for each shuffle mask, shuffling data from a data element designated by said shuffle mask to an associated resultant data element position if its flush to zero field is not set and placing a zero into said associated resultant data element position if its flush to zero field is not set.

7. (Original) The method of claim 6 wherein a capacity of a single register is 128 bits.

8. (Original) The method of claim 7 wherein each data element is a byte wide and each shuffle mask is a byte wide.

9. (Original) The method of claim 8 wherein said lookup mask is 128 bits long and L is less than seventeen.

10. (Original) A method for table lookup comprising:

loading data for a first M-bits wide portion and data for a second M-bits wide portion

of a table; loading an M-bits wide mask, said mask comprised of N control elements, each control element corresponding to a unique data element position;

shuffling said first M-bits wide portion in accordance to said M-bits wide mask to generate a first shuffled result;

shuffling said second M-bits wide portion in accordance to said M-bits wide mask to generate a second shuffled result;

merging selected data elements from said first and second shuffled results to obtain an M-bits wide table lookup resultant.

11. (Original) The method of claim 10 wherein said table and said portions of said table are comprised of packed data elements.

12. (Original) The method of claim 11 wherein said first M-bits wide portion, said second M-bits wide portion, and said M-bits wide table lookup resultant are each comprised of N packed elements.

13. (Original) The method of claim 11 wherein M is 128 and N is 16.

14. (Original) The method of claim 12 wherein each control element is comprised of: a flush to zero field, said flush to zero field to indicate whether a data element position associated with this control element is to be filled with a zero value; a selection field, said selection field to indicate which table data element to shuffle data from; and a source select field, said source select field to indicate which of said plurality of table sections to

shuffle data from for this control element.

15. (Original) The method of claim 10 further comprising generating a table select mask from M-bits wide mask, said table select mask to indicate which table section each resultant data element position should receive data from.

16. (Original) The method of claim 15 further comprising: applying said table select mask to said first shuffled result, wherein a first shuffled data element is selected from said first shuffled result; and applying said table select mask to said second shuffled result, wherein a second shuffled data element is selected from said second shuffled result.

17. (Original) The method of claim 16 wherein said merging selected data elements comprises merging data from said first shuffled data element and said second shuffled data element into said M-bits wide table lookup resultant, data from said first and data from said second shuffled data elements are to each occupy a separate data element position.

18. (Original) The method of claim 10 further comprising determining whether said table for said table lookup can fit into a single register, where if true, performing said table lookup with a shuffle operation on said table with said M-bits wide mask instead of performing lookups on multiple portions of said table.

19. (Original) The method of claim 18 wherein said single register is a 128 bit wide single instruction multiple data register, M less than 129, and said table is less than 129 bits wide.

20. (Previously Presented) An article comprising a tangible machine readable medium that stores a program, said program being executable by a machine to perform a method comprising:

determining whether a table having a set of L data elements fits into a single register;

performing a data lookup into said table with a packed data shuffle operation if said determination indicates that said table does fit into a single register; and

dividing said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and executing a plurality of packed data shuffle operations on said plurality of sections to look up data in said table.

21. (Original) The article of claim 20 wherein said method further comprises loading a lookup mask for each packed data shuffle operation, said lookup mask to indicate which data elements are to be extracted from said table.

22. (Original) The article of claim 21 wherein said lookup mask is comprised of L shuffle masks, each shuffle mask corresponding to a unique data element position.

23. (Original) The article of claim 22 wherein each shuffle mask is comprised of: a flush

to zero field, said flush to zero field to indicate whether a data element position associated with this shuffle mask is to be filled with a zero value; a selection field, said selection field to indicate which table data element to shuffle data from; and a source select field, said source select field to indicate which of said plurality of table sections to shuffle data from for this shuffle mask.

24. (Original) The article of claim 20 wherein said program further comprises merging shuffle results from said plurality of packed data shuffle operations into a single instruction multiple data register.

25. (Original) The article of claim 23 wherein each packed shuffle operation comprises: for each shuffle mask, shuffling data from a data element designated by said shuffle mask to an associated resultant data element position if its flush to zero field is not set and placing a zero into said associated resultant data element position if its flush to zero field is not set.

26. (Original) The article of claim 25 wherein each data element is a byte wide and each shuffle mask is a byte wide.

27. (Original) The article of claim 26 wherein said single register has a capacity of 128 bits and L is less than seventeen.

28. (Original) An apparatus comprising:

an execution unit to execute a sequence of instructions, said instructions to perform a table lookup operation, said instructions to cause said execution to:

determine whether a table having a set of data elements fits into a single register;

perform a data lookup into said table with a packed data shuffle operation if said determination indicates that said table does fit into a single register; and

divide said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and execute a plurality of packed data shuffle operations on said plurality of sections to look up data in said table.

29. (Original) The apparatus of claim 28 wherein said instructions are to further cause said execution unit to load a lookup mask for each packed data shuffle operation, said lookup mask to indicate which data elements are to be extracted from said table.

30. (Original) The apparatus of claim 29 wherein said lookup mask is comprised of a plurality of shuffle masks, each shuffle mask corresponding to a unique data element position.

31. (Original) The apparatus of claim 30 wherein each shuffle mask is comprised of: a flush to zero field, said flush to zero field to indicate whether a data element position associated with this shuffle mask is to be filled with a zero value; a selection field, said selection field to indicate which table data element to shuffle data from; and a source select field, said source select field to indicate which of said plurality of table sections to shuffle data from for this shuffle mask.

32. (Original) The apparatus of claim 31 wherein said execution is to comprises merging shuffle results from said plurality of packed data shuffle operations and to store said merged shuffle results into a single instruction multiple data register.

33. (Original) The apparatus of claim 31 wherein each packed shuffle operation comprises: for each shuffle mask, shuffling data from a data element designated by said shuffle mask to an associated resultant data element position if its flush to zero field is not set and placing a zero into said associated resultant data element position if its flush to zero field is not set.

34. (Original) The apparatus of claim 33 wherein each data element is a byte wide and each shuffle mask is a byte wide.

35. (Original) A system comprising:

a memory to store data and instructions; a processor coupled to said memory on a bus, said processor operable to perform instructions for a table lookup algorithm, said processor comprising:

a bus unit to receive a sequence of instructions from said memory;

an execution unit coupled to said bus unit, said execution unit to execute said sequence, said sequence to cause said execution unit to:

determine whether a table having a set of data elements fits into a single register; perform a data lookup into said table with a packed data shuffle operation if said

determination indicates that said table does fit into a single register; and

divide said table into a plurality of sections if said table does not fit into a single register, each of said sections sized to fit into a single register, and execute a plurality of packed data shuffle operations on said plurality of sections to look up data in said table.

36. (Original) The system of claim 35 wherein said instructions are to further cause said execution unit to load a lookup mask for each packed data shuffle operation, said lookup mask to indicate which data elements are to be extracted from said table.

37. (Original) The system of claim 36 wherein said lookup mask is comprised of a plurality of shuffle masks, each shuffle mask corresponding to a unique data element position, and wherein each shuffle mask is comprised of: a flush to zero field, said flush to zero field to indicate whether a data element position associated with this shuffle mask is to be filled with a zero value; a selection field, said selection field to indicate which table data element to shuffle data from; and a source select field, said source select field to indicate which of said plurality of table sections to shuffle data from for this shuffle mask.

38. (Original) The system of claim 37 wherein each packed shuffle operation comprises: for each shuffle mask, shuffling data from a data element designated by said shuffle mask to an associated resultant data element position if its flush to zero field is not set and placing a zero into said associated resultant data element position if its flush to zero field is not set.

39. (Previously Presented) The system of claim 38 wherein said execution unit is to merge shuffle results from said plurality of packed data shuffle operations and to store said merged shuffle results into a single instruction multiple data register.

40. (Original) The system of claim 39 wherein each data element is a byte wide and each shuffle mask is a byte wide.

IX. Evidence Appendix: With Copies of Evidence Relied Upon by Appellant

Appellant relies upon no additional evidence in this appeal.

X. Related Proceedings Appendix: Copies of Decisions Rendered by a Court or the Board in any Prior and Pending Appeals, Interferences or Judicial Proceedings

There are no related appeals or interferences to appellant's knowledge that would have a bearing on any decision of the Board of Patent Appeals and Interferences.